RiverCure Portal: Exploring Geographic Features on Context Definition and Integration with the HiSTAV Hydrometric Tool

Jorge Miguel da Silva Marques
Instituto Superior Técnico,
Universidade de Lisboa
Lisboa, Portugal
jorgemmarques@tecnico.ulisboa.pt

ABSTRACT

Flood events are becoming more prominent every day, causing serious problems to people affected by them, such as physical, psycologycal and material harm. With the advancement of technology, hydrometric modeling tools capable of predicting floods and quantifying their severity, like HiSTAV are emerging and becoming more proeminent. However, the dissemination of the results and the user interaction with these tools is still lacking. By taking advantage of the advancement of web and GIS technologies, an opportunity to solve these aspects arises. The objective of this thesis is to develop RiverCure Portal (RCP), a web application that, by integrating HiSTAV in its workflow, makes it widely available to knowledgeable users, disseminates its results to the interested stakeholders, and streamlines the user interactions necessary to use the tool. This development follows the latest software development methodologies like Model Driven Engineering (MDE) and uses recent technologies like Django, a Python framework for web development, and GIS libraries and databases. Moreover, RCP data model was specified in Application Specification Language (ASL), an ITLingo language focused on simplifying the development process by defining rigorous and platform-independent specifications. This thesis details the motivation behind RCP, and how these methodologies and technologies were leveraged to create a web application that integrates HiSTAV, in order to simplify its usability and disseminate its simulation results. A detailed walkthrough on how to create a HiSTAV simulation from RCP is described accompanied by images from the user point of view. Finally, an evaluation and a conclusion are provided, with the former being focused on the comparison of RCP with another similar tool called Mike 21 and, the latter containing the main takeaways from the development of RCP and whether it fulfills its intended goals. This conclusion is followed by future improvements to polish RCP and turn it in a truly advantageous tool.

Author Keywords

RiverCure; Water Management; Geographic Information System (GIS); HiSTAV; Application Specification Language (ASL)

INTRODUCTION

Countries and their responsible organizations for the water management require an ever-increasing level of sophistication to protect and provide better conditions to their citizens. Moreover, with the advances in technology and improvement in data collection, the amount of available data is bigger than ever, allowing the creation of richer and more complex hydrometric models that analyze this data. However, to fully leverage the advantages provided by these models, it is necessary to build pipelines responsible for gathering the collected data and feeding it to the models so they can correctly produce reliable and valuable results, which should be then disseminated through different stakeholders.

In 2018 the **RiverCure** project was promoted to solve this problem. **RiverCure** is a research project supported by *Fundação para a Ciência e a Tecnologia* (FCT), developed by *Instituto Superior Técnico* (IST), *Instituto de Engenharia de Sistemas e Computadores, Investigação e Desenvolvimento em Lisboa* (INESC-ID) and *Agência Portuguesa do Ambiente* (APA). It addresses the issue of reducing uncertainty and improving forecasting capabilities of hydrodynamic and morphodynamical mathematical models for flood simulation, water resources management and habitat protection [1]. Profiting from recent advances in collecting and processing crowdsourced information produced by riverine communities and shared in internet sites and social media, RiverCure intends to contribute to the improvement of the forecasting capabilities of mathematical hydrometric models of rivers, like HiSTAV [2], by making an efficient and systematic use of curated crowdsourced and authoritative data through assimilation and calibration [3]. To bring these goals into fruition a web application called RiverCure Portal, or RCP for short, that enables the gathering and funneling of water data collected by riverine communities and crowdsourced data into HiSTAV, and share the results of the execution of this tool with the stakeholders, was proposed.

RCP is based on different systems (explored later in section 3) which were partially integrated or emulated during this research. These systems are (i) **SNIRH** (*Sistema Nacional de*

Informação sobre Recursos Hidrícos or Hydric Resources National Information System in English), a water resources monitorization system, that saves data in APA's databases and releases the information to the public in a web portal (https://snirh.apambiente.pt) [4]; (ii) **SVARH** (*Sistema de Vigilância e Alerta de Recursos Hídricos* or Hydric Resources Surveillance and Alert System in English), a subsystem of SNIRH that provides the hydraulic state of rivers and reservoirs (i.e., water levels, flow rate, stored volumes and water quality) and relevant meteorological information in real time, while also allowing the prediction of its possible evolution [5]; and (iii) **HiSTAV**, a modelling effort aimed at delimiting the critical tsunami inundation areas in an urban waterfront and to quantify the associated severity [6]. Moreover, RCP should also integrate the treatment of images obtained from crowdsourcing in social networks (e.g., Instagram) using machine learning (ML) techniques to automatically classify these images according to their depicted water level. However, this integration is not part of the scope of this research and shall be treated in future work.

Additionally, this research proposed to tackle the problem that the creation and support of web applications are becoming increasingly knowledge demanding, with the involved actors being confronted by increasing demands for improved quality, increased complexity of products and services, higher flexibility, shorter lead-times, etc. [7], resulting in an increased monetary and temporal costs for the involved companies and individuals. In order to solve this problem, this research takes the development of the web application a step further by first devising a specification of RCP's data model in ASL. Inserted in the ITLingo initiative [8], ASL is a platform-independent application specification language that allows the user to write application specifications, from which a functional program can be generated [9]. Using an ASL specification as a starting point for the RCP development, ensured that the development of the RCP resulted in artifacts aligned with the stakeholders wishes, as these models were defined based on the sharing of a common vision and knowledge among technical and non-technical stakeholders, thus facilitating and promoting the communication among them. Furthermore, they made the project planning more effective and efficient while providing a more appropriate view of the RCP system and, allowing the project control to be achieved according to objective criteria [10]. This development methodology is known as Model Driven Engineering (MDE) [11] .

BACKGROUND

The RCP was development followed Model Driven Engineering (MDE) and Web Engineering Practices (WE). It was specified in ASL, a language from the ITLingo Initiative and based on RSL and IFML. Additionally RCP data follows the Simple Feature Access (SFA or ISO 19125) [12] from the OGC standards, and the WaterML standard. Its features were implemented using Django [13] extended with GeoDjango and, with a PostGreSQL extended with PostGIS database management system (DBMS).

Model-Driven Engineering (MDE) is a software development paradigm, at its core is the use of abstract models to describe software aspects while systematically transform these models into more concrete ones up to executable code [14]. MDE raises the abstraction level of languages needed to develop software [14]. It shields software developers from the complexities of underlying implementation platforms [15] offering potential benefits to software engineering including improved productivity, portability, maintainability and interoperability [16].

Web Engineering is the application of systematic, disciplined, and quantifiable approaches to development, operation, and maintenance of web-based applications. It is both a pro-active approach and a growing collection of theoretical and empirical research in web application development [17].

The standard Interaction Flow Modelling Language (IFML) is designed for expressing the content, user interaction and control behavior of the front-end of software applications [18]. The objective of IFML is to provide system architects, software engineers, and software developers with tools for the definition of Interaction Flow Models that describe the principal dimensions of an application front-end: the view part of the application, made of view containers and view components; the objects that embody the state of the application, and the references to business logic actions that can be executed; the binding of view components to data objects and events; the control logic that determines the actions to be executed after an event occurrence; and the distribution of control, data, and business logic at the different tiers of the architecture [19].

ITLingo is a long term initiative with the objective of researching, developing and applying languages of rigorous specifications [20]. ITLingo approach has the objective of improving technical documentation, and the knowledge extraction and conversion in order to help stakeholders gaining a better understanding of software requirements, which are often misunderstood [21]. It brings three different languages for three different but close domains: Requirements Engineering, with RSL (Requirements Specification Language); Testing Engineering, with TSL (Testing Specification Language) [22]; and Project Management, with PSL (Projects Specification Language) [23].

RSL is a requirement specification language inserted in the ITLingo Project. It is a process and tool-independent language, meaning it can be used and be adapted by multiple users and organizations with different processes/methodologies and supported by multiple types of software tools. However, in practice, RSL has been implemented with the Xtext framework [24] in an Eclipse-based tool called "ITLingo-Studio" so, its specifications are

rigorously defined and can be automatically validated and transformed into multiple representations and formats[23]. RSL provides several constructs logically classified according to two dimensions (see Table 2) abstraction level and specific concerns they address. The abstraction levels are: business, application, software and hardware levels [23]. It allows the definition of DataEntities and DataEntitiesCluster, Actors, UseCases and StateMachines. It is important to understand these concepts once that ASL also uses them.

ASL [9] was built on top of RSL, (adding a third dimension) that allows the user to describe containers. To understand ASL it is important to first understand RSL. It is the most recent language integrated in ITLingo initiative. Like RSL, ASL has also been implemented with the Xtext framework [24] in an Eclipse-based tool called "ITLingo-Studio". So, its specifications are rigorously defined and can be automatically validated and transformed into multiple representations and formats. ASL uses concepts from both RSL and IFML which are domain specific languages (DSL). While RSL brings text-based requirements for what business software applications should consist of, IFML brings a visual representation of how the applications should be presented to and manipulated by the final users. ASL keeps RSL concepts like DataEntity, DataEntityCluster, Actors and UseCases.

The Open Geospatial Consortium (OGC) [25] is an international voluntary consensus standards organization, committed to improving access to geospatial, or location information [25]. It provides a consensus process that communities of interest use to solve problems related to the creation, exchange and use of spatial information. OGC standards are technical documents that detail interfaces or encodings [26]. The most relevant of these documents, for this research, is the Simple Feature Access (SFA), also known as ISO 19125. This standard focus on describing the common architecture for simple feature geometry, which is Distributed Computing Platform neutral and uses UML notation [12]. The RCP data model definition follows the SFA standards defined by this organization, in what concerns the geospatial information. Developing the RCP data model based on these standards increases the portability and potential cross-functionality with other systems.

WaterML 2.0 is a technical standard and information model for the representation of water observations data that, by using the existing OGC standards [26], aims at being an interoperable exchange format, allowing the exchange of water observations data sets across information systems [27]. It is divided in 4 parts. Part 1 is about timeseries, Part 2 about ratings gaugings and sections, Part 3 is about surface hydrology features and Part 4 about GroundWaterML. Since RiverCure intends to manage, collect, and distribute data coming from different sources, it is important to standardize data according to WaterML, as interoperability is an important requirement.

To define a location on the planet, it is necessary to use a coordinate reference system (CRS). With the help of CRS, every place on the earth can be specified by a set of two or three numbers (2D or 3D), called coordinates. has 5 key components [28]: **Coordinate system:** The X, Y grid upon which your data is overlayed and how you define where a point is located in space; **Horizontal and vertical units:** The units used to define the grid along the x, y (and z) axis; **Datum:** A modeled version of the shape of the Earth which defines the origin used to place the coordinate system in space; **Projection Information:** The mathematical equation used to flatten objects that are on a round surface (e.g., the Earth) so we can view them on a flat surface (e.g., your computer screens or a paper map). These approaches are designed to increase the accuracy of the data in terms of length and area. Individual CRS can be referred to using Spatial Reference Identifiers (SRID) integer, including EPSG codes defined by the International Association of Oil and Gas Producers (IOGP).

RCP handles GIS data which can be divided into the following types: (i) Vector data is a way to represent real world features within a GIS environment. Things like roads, trees, rivers can be represented by a vector feature. Vector features are shapes represented by geometries such as point, line and polygon [29], with every point coordinate value being dependent on the CRS in use; (ii) a Raster is a matrix of cells (i.e., pixels) organized into rows and columns (i.e., a grid) where each cell contains a value representing a variable, such as temperature or height [30] (example on Figure 12). Rasters are well suited for representing data that changes continuously across a landscape (surface). A digital terrain model (DTM) is a popular example of a raster representation. DTMs are an important input for both the HiSTAV Simulation pipeline and the user of RCP. (iii) a Mesh is an irregular triangle net, usually with temporal and spatial components. It provides information about the spatial structure that contains a collection of vertices, edges and faces in 2D or 3D space [31] and, can have datasets that assign a value to every vertex. A vertex is a XY(Z) point, an edge is a line connecting two vertices and, a face is a set of edges forming a closed shape. Each vertex can store different datasets with or without a temporal dimension [31], containing information like, for example, wind speed through time.

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design [32]. It is focused in developing web application following a Model-View-Controller (MVC) [33] architecture and has 3 main principles [13]: Don't repeat yourself (DRY), explicit is better than implicit and, loosely coupled architecture. GeoDjango [34] is an included contributed module for Django [32] that turns it into a world-class geographic web framework. Since RCP needs to include geographic functionalities and deals with different data formats, GeoDjango functionalities integration is a right decision. Moreover, GeoDjango strives to make it as simple as

possible to create geographic web applications like location-based services [35]. Its features include: (i) Django model fields for OGC geometries and raster data; (ii) extensions to Django's ORM for querying and manipulating spatial data; (iii) a loosely coupled, high-level Python interfaces for GIS geometry and raster operations and data manipulation in different formats.

PostGreSQL DBMS is a powerful, open source object-relational database system with over 30 years of active development that has earned a strong reputation for reliability, feature robustness, and performance [36]. Moreover, it has an extension, known as PostGIS [37], that has a vast set of geographic operations based on the OGC standards (e.g., CRS transformations, geometries intersection) [38].

RELATED WORK

RCP is, in essence, an integration and streamlining of certain features and functionalities of systems in use today. There are 3 main systems that need to be understood, **SNIRH** [4], **SVARH** [5], and **HiSTAV** [2]. The first two serve both as a feature requirement collection and a data collection location. When it comes to features, these systems have functionalities that were emulated in RCP like the visualization of sensors on the map, where the user can collect all the information about the sensors by clicking on them. In terms of data, these systems have a vast network of sensors spread around Portugal that collect hydrometric data. These sensors should be integrated with RCP in the future, allowing the flow of data from these sensors to RCP. As for **HiSTAV**, a system that simulates a Context design on RCP, it is fully integrated with RCP, allowing the full leverage of all its functionalities, in a way that is imperceptible for the user (i.e., the user does not interact with **HiSTAV** directly).

Moreover, to validate RCP and its utility, it is necessary to compare its features with existing tools available in the market. One of such tools is Mike Powered by DHI [39]. Mike offers several suits to solve different problems, but in the scope of this research, the main focus is on Mike 21 [40], as this suit is the most similar to HiSTAV. In this section a detailed description of the aforementioned tools is provided.

SNIRH

Sistema Nacional de Informação sobre Recursos Hidrícos (SNIRH) or Hydric Resources National Information System in English, was announced on October first ,1995 by *Instituto da Água* (INAG) and is a water resources monitorization system, that saves data in APA's SQL-Server DB and releases the information to the public in a web portal (https://snirh.apambiente.pt). The portal gets 600 visits per day, between teachers, students, researchers, journalists and public administration personal and is divided in three sub-systems, SNIR-Lit, SNIRH-Júnior and, SVARH [4].

The web portal shares relevant information, like reports and maps of flood areas. However, it is quite old, and the user interaction is poor. RCP emulates the most relevant functionalities of SNIRH, mainly the display and availability of the vast amount of information collected in APA's SQL DB, while significantly improve the usability for the user. RCP then further expands on SNIRH's concept by integrating a hydrometric model known as HiSTAV (explored in section 3.3) creating the ability to perform simulation through interaction with the portal, strengthening the amount of data available on the RCP.

In summary, SNIRH's concept is the base from which RCP was built. Both systems share a great deal of requirements and functionalities. The main differences being the improvement of user interaction, visual appeal and the integration of HiSTAV in RCP.

SVARH

Sistema de Vigilância e Alerta de Recursos Hídricos (SVARH) or Hydric Resources Surveillance and Alert System in English is a subsystem of SNIRH that provides the hydraulic state of rivers and reservoirs (i.e., water levels, flow rate, stored volumes and water quality) and relevant meteorological information in real time, while also allowing the prediction of its possible evolution. It is composed by a network of stations with autonomous transmissions, that measure several hydrological variables and water quality, and by a technology infrastructure for storage and dissemination of the data collected by the stations.

This network is composed by three types of automated stations 311 hydrometric and 620 meteorological, totalizing approximately 931 stations [41]. The data collected by each station is related with the station type (e.g., meteorological measures water level, wind speed and direction, air temperature, relative humidity, and water temperature). A station is composed by sensors, a datalogger, a power supply (battery and solar panel), and a communication system (usually GSM modem) [42]. The flow of the collected data between these components is seen on Figure 16. These stations are located in critical points for surveillance of floods, droughts and pollution accidents [5]. The whole system is divided in three parts: (i) **data acquisition** – Automatic stations with transmission; (ii) **central processing** – Data gathering informatic system from the automatic stations, and its storage with hydric and hydraulic models; and (iii) i**nformation distribution** – Real-time information distribution software from the automatic stations.

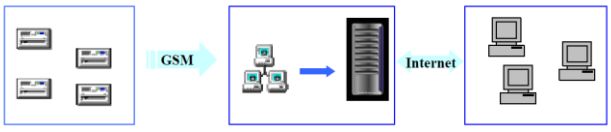

Figure 1. From left to right: Data Acquisition, Central Processing and Information Distribution (Extracted from [5]).

The main relevance of this system in RCP context is the possibility to leverage the data distribution of SVARH and funnel the data to RCP. Making it possible for RCP's stakeholders to have more precise and trustworthiness results, as well as more distinct Contexts available on RCP (i.e., more area is covered/protected by RCP).

HiSTAV

HiSTAV is a modelling effort aimed at delimiting the critical tsunami inundation areas in an urban waterfront and to quantify the associated severity [6]. Concretely, it is a reliable and performant tool for faster than real time (FTRT) high-resolution simulations, leveraging parallel, distributed and graphics-based computing technologies for this effect. Moreover, HiSTAV was released as a standalone product, available to both engineering and research communities, by supporting a close integration with open-source Geographic Information Systems and scientific visualization toolkits for complex data handling and analysis. Its applicability remains valid for multiple fields in water resources [2]. For its simulation, HiSTAV needs a collection of geometries and rasters known as RCP Context as input.

RCP integrates this tool by implementing features that allow a user to define or upload the Context described above, as well as its corresponding properties, and establishing a pipeline between itself and HiSTAV enabling bidirectional data exchange. This pipeline allows RCP users to execute simulations based on defined "Contexts" and visualize the results.

Mike 21

Mike 21 is a versatile desktop tool for 2D coastal and sea modelling. It is capable of simulating physical, chemical, and biological processes in these contexts. The hydrodynamic model in the MIKE 21 Flow Model is a general numerical modelling system for the simulation of water levels and flows in estuaries, bays and coastal areas. It simulates unsteady two-dimensional flows in one layer (vertically homogeneous) fluids and has been applied in a large number of studies [43]. It is capable of considering different flood factors like **Boundary Conditions**, **Flooding and Drying**, **Infiltration and Leakeage**, and **Multi-Cell Overland Solver**. Additionally, it contains a **Flood Screening Tool** [43]. It has a proven track record as it has been used for many costal and marine engineering projects around the world [40]. Some of its benefits and typical applications, according to its developer, are [40]: Benefits: (i) proven technology and more than a 25-year track record of successful applications; (ii) offers maximum flexibility, higher productivity, and full confidence in the results; (iii) is modular; (iv) it comes with a wealth of first-class tools that enhance and ease modelling possibilities; In terms of typical applications: (i) design of data assessment for coastal and offshore structures; (ii) optimisation of port layouts and coastal protection measures; (iii) cooling water, desalination, and recirculation analysis; (iv) optimisation of coastal outfalls; (v) environmental impact assessment of marine infrastructures; (vi) ecological modelling including optimisation of aquaculture systems; (vii) optimisation of renewable energy systems; (viii) water forecast for safe marine operations and navigation; (ix) coastal flooding and storm surge warnings; (x) inland flooding and overland flow modelling;

As for the user interaction, to correctly use Mike 21, the user needs a background in coastal hydraulics and oceanography, which is sufficient to check whether the results are reasonable or not [44]. The general steps taken to simulate an area in Mike 21 are [45]: (i) the bathymetry needs to be setup by importing all the necessary geographical data with soundings based on a survey or digitized from nautical chart; (ii) the user needs to create the boundary conditions by setting the water levels at the boundaries; (iii) define a data set with all the values to simulate (e.g., water level through time); Figure 17 shows an example of a Mike 21 working area.

There are several similarities between Mike 21 and RCP, both in user qualifications, general guidelines to simulate an area and, business goals. Chapter 6 provides a detailed comparison between the two tools.

RIVERCURE PORTAL REQUIREMENTS

This research only covers a part of the entire RCP system. To have a better understanding of what was developed in this research and its purpose, it is necessary to have a better idea of the entire RCP system. For this effect, the RCP requirements are detailed in this chapter followed by a description of what part of RCP was implemented in the scope of this research.

The RCP intends to integrate SNIRH and SVARH functionalities with HiSTAV itself in one web application [46]. As such, there are four main points that need explaining. First, the RCP interface, i.e., which interactions should be available to the user and the general structure of the data visualization. Second, the definition of the Context concept, which is the main focus of this research and is the input for HiSTAV simulations. Third, how data is collected and funneled to RCP, as well as what RCP does with it. Fourth, how RCP should integrate HiSTAV into its workflow, and what are the data flows between these two systems. Additionally, a last section is provided detailing which components were developed within this research project.

RiverCure Portal Interface

RCP interface is inspired on SNIRH [4] interface. However, it intends to significantly improve the user interaction as well as the overall appearance of the site. RCP relies on the segregation of responsibilities between users according to their specific roles (e.g., a visitor will not have access to the HiSTAV simulation request but, a Context Admin will). In terms of user interaction, this segregation is leveraged by providing a custom interface for each user role (i.e., each navigation button is only available to a user which role can use the associated functionality).

The information and functionalities that should be available in RCP are the following: (i) information about all the created “Contexts”, and a set of actions to create such “Contexts”; (ii) information about all the sensors associated with RCP system, and functionalities to associate new sensors. The sensor information should contain the details of the sensor itself (e.g., code, location, type) as well as the data being collected, in the form of observations; (iii) information about past, present and future flood events, as well as a way to create such events; (iv) full integration with HiSTAV, allowing the user to use its functionalities without interacting with it directly. Additionally, RCP should contain a similar view to the one on the left of Figure 2.

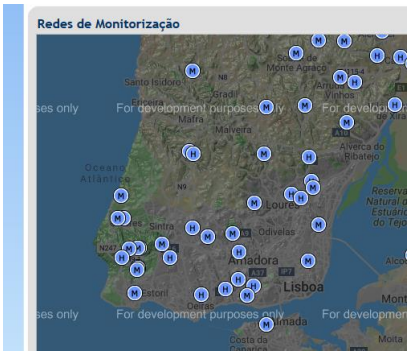

Figure 2. SNIRH system monitorization network (Extracted from [4]).

RiverCure Portal Context

The Context concept is composed of several features, which are divided into 2 different groups: **Context Geometries** and **Context Attributes**. **Context Geometries** are composed of distinct geometries named: Domain, Refinement, Alignment, Boundary and Boundary Point. On the other hand, **Context Attributes** are: DTM Raster, Context Sensor, Hydro Feature, and Context Event. Spec. 5 shows the rigorous Context specification in the ASL language. The first 5 concepts, under the geometries section, are geometries that, have a visual representation, and need to be defined or imported by the user on RCP. These concepts are known by HiSTAV as Context files and, are the core input for its simulation. The other 4 are either defined through associations of concepts (e.g., associating a sensor to a Context, in the case of the Context Sensor, or associating an Event to a Context, in the case of Context Event), are an imported file (DTM Raster) or were defined previously and are transversal to the entire system (Hydro Features). Except for the DTM Raster, whose file representation is needed by HiSTAV, all the other three concepts are not used directly by HiSTAV and are instead used by RCP to configure a simulation request.

```
DataEntity e_Context "Context": Master [
        attribute id "Id": Integer [constraints
(PrimaryKey)]
        attribute code "Code": String
[constraints(NotNull Unique)]
        attribute Name "Name": String
[constraints(NotNull Unique)]
        attribute hydroFeature "HydroFeature": Integer
[constraints(NotNull ForeignKey(e_HydroFeature))]

        attribute geomExternalBoundary "Domain"
GeoPolygon
        attribute CLExternalBoundary "Domain CL": Double
        attribute geomRefinement "Refinement":
GeoPolygon [constraints(multiplicity"1..*")]
        attribute CLRefinement "Refinement CL": Double
[constraints(multiplicity"1..*")]
        attribute geomAlignment "Alignment": GeoPolyline
[constraints(multiplicity"0..*")]
        attribute CLAlignment "Alignment CL": Double
[constraints(multiplicity"1..*")]
        attribute userResponsible "User Responsible":
Integer [constraints(NotNull ForeignKey(e_User))]

        attribute isPublic "Context Access
Restrictions": Boolean [defaultValue "False" constraints
(NotNull)]
```

Spec. 1. Context ASL specification.

Context Geometries

The definition of these geometries is the bulk of the work related to the definition of a Context. These geometries represent the visual part of a Context. When the user is visualizing a Context what the user is seeing is these 5 geometries definition. The DTM also has a visual representation within a Context, however, since it is not a geometry, but a raster, it cannot be defined by the user, only uploaded in a raster file format form, and is treated differently by HiSTAV, it is under a different section as it is not considered primary in a Context definition. In a structural sense, these geometries are similar because they follow the same design principles, as they are all geometries and can be defined by the user by drawing on a map. These geometries are stored in a PostGIS DB [37] and have a corresponding geojson file [47] representation, which is not stored but generated on demand. Lastly, the definition of these geometries is subject to 2 geographic constraints: (i) the Domain geometry must contain, inside its boundaries all the other geometries; (ii) the boundaries of these geometries cannot intersect each other under any circumstance.

1. Domain

The layman definition of Domain is that of a general area which is under analysis for the potential risk of flood. In most cases, this is the first geometry defined by the specialized user. A Domain geometry is a **Polygon**. Each Context can only have 1 Domain that is defined by a single Polygon. This geometry only has 1 property, cell length (CL), which is a Float, and is defined by the user, after defining the Domain Polygon.

2. Refinement

A Refinement geometry is a Polygon. It must be contained inside the Domain geometry and, it is usually defined around the area of interest of a river, resulting in said river being contained inside the polygon. However, rivers can have tributaries, or other features, requiring more than 1 polygon to be represented as a correct pre-processor geometry and input, each with individual property values. The only property of this geometry is also named CL and, in case there

are several refinements defined for the same Context, each refinement has its own CL value.

3. Alignment

An Alignment is defined by a LineString (GeoPolyline in ASL). It must be contained inside the Domain and, it usually outlines a river. It might be necessary to define several alignments for a Context due to the existence of tributaries and other features of a river. These definitions cannot intersect each other. The definition of alignments is not mandatory, as long as the refinements have a sufficient level of detail to form the Mesh grid. The only property of the alignment geometries is CL, defined by the user after defining the geometry points.

4. Boundary

A Boundary is defined by a LineString. Its definition must be overlaid on the Domain boundaries and all its points must be points from the Domain definition. Consequently, it is defined by joining at least 2 sequential points from the Domain. In almost all cases a Context will have more than 1 Boundary, the user can therefore define several Boundaries for a Context as long as they don't overlap or share any points. Each Boundary has 2 properties, the Type and the Data Type. Both properties are multiple choice fields and are defined by the user after defining the boundaries. The Type can be either Input, Output or InputOutput. Regarding the Data Type the choices are H for depth, Q for discharge, Z for elevation and V for velocity.

5. Boundary Point

A Boundary Point is a point from a Boundary. These are automatically created when the user defines a Boundary, 1 Boundary Point for each point of the Boundary. A Boundary Point only has 1 property named series, which can be defined multiple times. This property represents an association of a boundary point to a sensor, which becomes a Context Sensor after the fact (Context Sensor concept is explained next, in the Context Attributes section). The association is done by selecting a sensor, which is within a defined distanced to the Boundary, from a boundary point popup interface. The distance for possible sensors association is also defined in this popup interface.

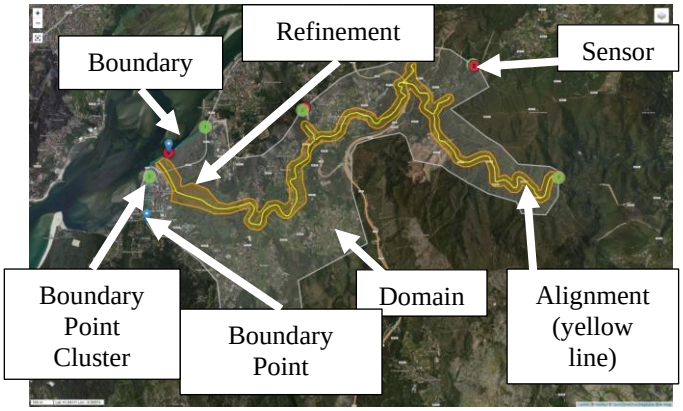

Figure 3. Context Visualization on RCP.

RIVERCURE PORTAL – SIMULATION PIPELINE

ASL Data Types and Mapping

After analyzing the objectives and goals of RCP four spatial data types, based on OGC standards [12], were chosen. These data types are generalist to provide the necessary flexibility for the development of a web application of this kind, as well as allowing the exploration of ASL geographic code generation mechanisms for other programming languages in the future, since ASL is a platform-independent language. These types were named: **GeoPoint**; **GeoPolyline**; **GeoPolygn**; **GeoRaster** and, were mapped as seen on Table 1.

Table 1. Datatypes mapping from ASL to Django (Python)**.**

ASL Data Type	GeoDjango Data Type
GeoPoint	PointField
GeoPolyline	MultiLineStringField
GeoPolygn	MultiPolygonField
GeoRaster	RasterField

Context Definition

The user can select a Context to define by selecting one from the Context list page, accessible from the landing page of RCP. If the user does not find the intended Context on this list, he can either request access to it, if the Context is present in RCP but not accessible to the user (these Contexts are found by searching other Context list page, accessible from the Context list page), or he can create a new one with the intended properties (name, code, and hydro feature). After selecting the Context to edit the user is redirected to a Context detail page similar to the one on Figure 4. Here, at the bottom of the page, the user has several options, we will go through each one of them in order.

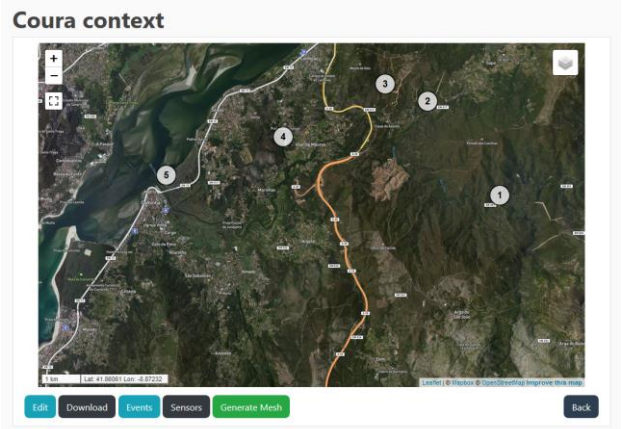

Figure 4. Context detail page view.

The edit option allows the user to edit the geo features of the Contexts and change its name or hydro feature. Clicking on this button prompts a form where the user can upload possible Context geometries, a DTM, or/and Context Countour Lines files he might have. For the geometries these files have to be in the geojsons format and for the other two the TIFF format. The user does not need to upload the entirety of the files and can choose to just upload a select few. The user must then define manually the files which, were not uploaded, in order to perform a HiSTAV simulation. The DTM and Contour Lines files, can only be uploaded and not defined manually. Choosing the manual edit option in the prompt redirects the user to a page where he can draw the geometries on a map similar to the one on Figure 4. He just needs to identify which geometry he is drawing and attribute it its properties, either by clicking on the geometry and inserting the value on a popup that opens on click, or by inserting the value directly on a geometry tree present on the same page, that contains all the drawn geometries. After drawing the geometries, the user just needs to validate and save the Context. There are some controls in place to guide the user to a complete and valid Context definition, like an alert informing which geometries are still missing, requiring the Domain geometry to be the first to be drawn as all the other geometries must be contained inside this one, and visual information when the geometry being drawn is intersecting itself. Additionally, sensors and boundary points are clustered (each on its own group) to avoid cluttering the map with information.

The download option allows the user to download all the defined geometry files in geojson format. Only the defined geometry files are provided in the download (e.g., if only the Domain and refinements are defined, only two files are provided containing the respective information.

The events and sensors option redirect the user to the events and sensor list pages, respectively, associated with the Context. On the event list page the user can create a new Context event that triggers a request to HiSTAV for the respective Context. The necessary parameters are defined by the user on a form filled on event creation. These parameters are transformed in text files that HiSTAV can read. Before creating this event, it is necessary to generate the Context mesh explained next.

The generate mesh option triggers a request to HiSTAV for a mesh generation of the defined Context. Clicking the Generate Mesh button RCP transforms the Context Geometries in several different serialized geojsons, one file for each geometry type, that are sent in the request body. The user receives a response from the other application indicating whether the pre-processing request was successful or not. The pre-processing is an asynchronous operation due to efficiency (the pre-processing takes some time), so the user does not get immediate feedback of whether the pre-processing was successful or not. Instead, the other application will notify RCP when the mesh generation is finished. This notification is done through a RCP endpoint. This option changes its color to yellow and text to regenerate mesh in case the mesh was previously generated.

HISTAV INTEGRATION

HiSTAV simulation pipeline consists of 2 phases. Firstly, a pre-processing of the Context is necessary, followed by the simulation using a software called solver2D. In the pre-processing phase, a user needs to make a request and provide the 5 geometries files: (i) Domain; (ii) refinement; (iii) alignment; (iv) boundary; (v) and boundary points. This is done by simply clicking a button as seen on the previous Pre-processing request section. These files are used by a software called mesh to generate a raster (example shown on Figure 39), saved in VTK format, which is an input for the solver2D software. From this point onward this mesh software will be addressed as pre-processor. After this VTK file is created, the Context is ready for the simulation by solver2D (explained in section 6.2). It is possible for the user to verify and validate the output of mesh by using Paraview Web Visualizer. Although it is not part of the simulation pipeline directly, it is an important validation tool, and, as such, it is addressed in this section.

The executions of the pre-processor and solver2D are independent of each other. This means that, in theory, the execution of the solver2D does not have to be preceded mandatorily by the execution of the pre-processor. However, since the output of the pre-processor is an input for the solver2D, in practice, the pre-processor needs to be executed at least once before executing the solver2D. After generating the mesh, solver2D can be execute as many times as needed with different constraints (e.g., time intervals or frequency outputs), without the need to execute the pre-processor again, as long as the Context definition does not change. A simplification of this pipeline can be seen on Figure 4.

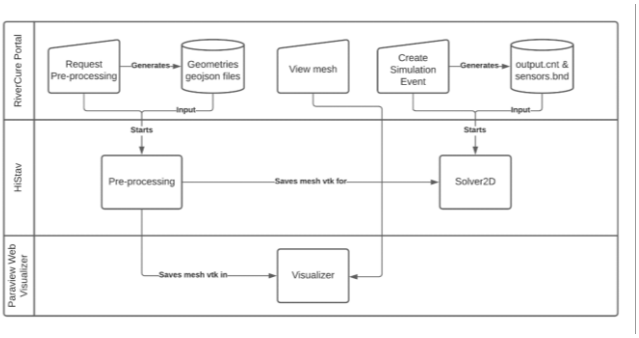

Figure 5. HiSTAV integration with RCP.

Pre-processor

The objective of the pre-processor is to generate a mesh called **meshQuality.vtk**. This VTK file is an input for a Solver2D execution is requested. Without this file, Solver2D cannot execute successfully and, as such, our value stream is compromised. As a consequence, the end-user should always verify if a given Context has an updated mesh before requesting a simulation for such Context. The pre-processing inputs and output are the following: **The Geometry files** are

geojson files that contain the geo-information and its properties of the several geometries designed on RCP and analyzed in the previous section. Each file contains the information corresponding to its name. Since we already went over these geometries in detail and their geojson representation on the previous section, 5.2 Context definition, we refrain from doing it again here. An enumeration of the different files follows: (i) domain.geojson; (ii) alignments.geojson; (iii) refinements.geojson; (iv) boundaries.geojson; (v) boundaries_points.geojson; **The DTM** is a file, called dtm.tif, that contains a raster in geoTIFF format known as the digital terrain model or for short, DTM. The end-user is responsible for uploading this file to RCP, and it should correspond to an area overlapping and slightly larger than the Domain geometry area; **The Friction Coefficient** is a file, called frictionCoef.tif, that contains a raster in geoTIFF where each pixel has the friction coefficient value of the terrain located in the same coordinates as the pixel; **The output of the pre-processor** is a mesh file, containing a triangular grid, corresponding to the Context given as input.

Paraview Web Visualizer.

As we have seen, before performing the HiSTAV simulation, the user needs to perform a pre-processing task to generate a mesh for the Context under simulation. This mesh is generated and saved in a VTK format and is an important input for the HiSTAV simulation. As such, the user might want or need to check if it was correctly generated before starting a simulation. Hence a tool that allows the visualization and interation with such a file was necessary. For this project Paraview Web Visualizer [48] was chosen as it provides all the necessary features for the validation of the mesh.

Solver2D

Solver2D is the software responsible for simulating a given Context. The execution is the last step on the value stream this project set out to achieve. Solver 2D has two inputs, which are named sensor.bnd(s) and output.cnt, and a series of outputs. A description of each input and output follows: **Sensor bnds** are several text files, each corresponding to a certain sensor associated with a Context point of the Context under simulation. In order to distinguish to which sensor belongs each file, the files are named after the sensors (e.g., sensor_mare.bnd, where mare is the name of the sensor). The file data is comprised of 2 columns, separated by a tab. The column on the left contains the instant corresponding to a value on the right of a reading by the sensor, of for example, a discharge. Every file left column starts on instant 0 adding 60 seconds per each row until reaching the duration of the simulation. The instant 0 is the instant of the beginning of the simulation and contains the value of the sensor observation with the corresponding timestamp, e.g., if the simulation was configured to start at 01/12/2021 at midday, then the value on the right of 0 would be the sensor observation with the date 01/12/2021 and time 12:00. The same principle applies for all the other values. The start and end datetime of the simulation are defined by the user on RCP when creating a simulation event through a form (seen in section 5.2 Context Event Simulation). Figure 40 shows an example of a complete Sensor bnd file contents. **Output.cnt** file is comprised of two lines. Each line contains one value in hz. The first line corresponds to the frequency at which the user wants to write to the files being generated. The second line corresponds to the frequency at which the user wants to update the maximums during the simulation. **Solver 2D Output** is a VTK file. An example can be seen on Figure 41. A specialized user can analyse this file to assess the risk of flood for the Context Event from which the Simulation resulted. Additionally, there is the possibility of transforming this VTK into the following four different TIFFs: (i) max depth; (ii) max level; (iii) max q; (iv) max vel. This transformation is made using a python script named stavResults.py that was not developed in the scope of this project. These TIFFs allow the specialized user to reach the same conclusion with the advantage of being significantly smaller in size, resulting in the decision to use these TIFFs instead of the VTK to increase performance. Additionally, during the simulation, other VTK files related with hydrodynamics are generated on HiSTAV. The number and size of these files is large and, dependent on the Writing Periodicity defined by the user during Context Event creation (i.e., the higher the frequency, the higher the number of files). This makes these files not suitable for display on RCP. As such, these files are only available to users with access to the HiSTAV machine. Possibly, in the future, these files could be made available on the Paraview Web Visualizer since they are VTKs.

Simulation API

This API serves as the communication medium between HiSTAV and RCP, and is built with Flask [49]. HiSTAV has three available requests, which are pre-processing, simulate and, simulation results request, while RCP has two endpoints that HiSTAV can use to communicate requests completions.

The Pre-processing endpoint enables RCP to request a mesh generation for a given Context. For this effect RCP must provide the geojson files representation of the five “Context Geometries”, one file for each geometry, on the request body. The request will return a string with either “success” or “fail” value, with the former representing a successful mesh generation process start and the latter the contrary. Since the mesh generation is done asynchronously, this request also makes a request of its own to the RCP. Its purpose is to signal RCP that the Context pre-processing has been finalized and has a valid generated mesh. This request is made using cURL [50] after the mesh program exits with a successful code. In case the processing is unsuccessful, no request is made.

The Simulate endpoint enables RiverCure to request a simulation for a given Context Event. For this effect RCP must provide a text file named output.cnt and the collection of bnds files with the Context Sensors Observations data.

The request will return a string with either "success" or "fail" value. The former representing a successful mesh generation process start and the latter the contrary. Since the mesh generation is done asynchronously, HiSTAV also makes a request of its own to the RCP to signal the end of the simulation using cURL. This request will trigger RCP to request HiSTAV the Simulation Results.

The Simulation Results endpoint provides the requester with a zip folder containing the 4 TIFFs generated from the maximum.VTK with the stavResults.py. The generation from VTK to TIFF is done after the request to this endpoint is made. The arguments serve as identifiers of which Context and Context Event belong the results being requested.

EVALUATION

Context Definition Evaluation

The first part of the evaluation was focused on the use case of the Context Definition. This is essentially the definition of the different required geometries. On Mike 21, this definition is done on the desktop application itself or ARCGIS. For RCP, a different approach was selected. The main idea was to join every operation in the same program from the user perspective, so that he only has to interact with one application (RCP). For this effect, every week, RCP features were implemented that fulfill the smaller use cases of the Context Definition, (e.g., Domain definition, Boundary definition…) until fulfilling 100% of the Context definition requirements. Tools like ArcGIS [51] or QGIS [52] can also be used to treat or define data to be uploaded to RCP, bypassing some of its limitations, but sacrificing some interaction streamlining (i.e., an additional step is added), or Mike 21. The implementation of these features was considered successful when the output from the Context Definition on ArcGIS and RCP produced a valid output for an HiSTAV simulation.

In the definition of auxiliary files (e.g., the DTM) RCP is severely limited, requiring the user to upload an already existing file, previously defined elsewhere. With Mike 21 the user uses ArcGIS, which does not have this limitation, as it is possible to define such files on ArcGIS. In RCP favor is the fact that it is a web application, contrary to ArcGIS which is a desktop application. This results in a high availability of RCP as it is accessible to anyone, with an authorized account, from anywhere where internet is available. It is also possible to easily share defined Contexts between users. Moreover, the introduction of new sensors in the system is more linear on RCP. Sensors can be added to the RCP with the corresponding data and associate such sensor to a Boundary Point in a streamlined way. In Mike 21 it is required for the user to find and associate the sensor data manually.

Simulation Evaluation

On the subject of completing a simulation the differences are as follows: On Mike 21, the user gets the output from the ArcGIS definition, consisting mainly of shapefiles and TIFFs. Additionally, the user must also get a timeseries file for each sensor associated with the Boundary Points and define some parameters like the writing of new maximums frequency. These steps are error prone as there are too many manual definitions which can result in an invalid simulation. On RCP, each sensor timeseries is automatically created, based on the time interval defined by the user when creating a simulation Event. Since each sensor was associated to a Boundary Point during the Context Definition phase, the association of each timeseries files to the corresponding geometry is done automatically. As for values like the frequency of writing of new maximums, they are also provided by the user on the simulation Event creation. From the user point of view this involves clicking 2 buttons and filling out a small form. We conclude that in RCP simulation pipeline the simulation execution process is much more streamlined than in Mike 21. This results in a more trustworthy result for each simulation as well as a much better user experience, since the user does not have to perform as many steps on RCP.

Overall, the differences between RCP and Mike 21 are considerable. Whereas Mike 21 provides a more complete set of simulations, RCP focus more on the user experience and dissemination of results in real time. A more detailed comparison between the two systems will be needed in the future, when RCP is further polished, with more users and rigorous Context definition use cases.

CONCLUSION

The necessity of tools such as RCP is more relevant than ever. However, there has been a lack of leverage of the web from existing tools. RCP successfully solves this problem. The fact that it is a web application with a PostGIS Database makes it easier to share and develop results and opens the door to the integrations of diverse data streams due to the use of OGC standards. Moreover, RCP streamlines the interaction of the user with Water Management tools by automating some crucial steps that had to be done manually in other tools. This results in an improved user experience, which results in a higher likelihood of the specialized user performing a good job.

It is important to not forget that, at the moment of writing, RCP is still in its first iteration, still missing some of its envisioned features like the constant stream of data to the Portal. A fully fledged RCP would constantly simulate, requiring user interference only when defining new rules or Contexts. A tool like RCP can change how governments and civil protection organizations protect their citizens, allowing these organizations to predict moderate to serious flood events. With these predictions these organizations can act in order to minimize damage and save lives. Additionally, it was also proved with this work that an ASL specification can generate a valid data model, from which an entire application can be developed. By having a transversal specification that can be understood by most stakeholders, like ASL, we are minimizing the risk that the software under development does not correspond to the expectations of all stakeholders.

REFERENCES

[1] "About – RiverCure Project." https://www.inesc-id.pt/projects/II11041/ (accessed Dec. 10, 2020).

[2] D. A. S. Conde, "High-Performance Modelling of Tsunami Impacts on Built Environments," Instituto Superior Tecnico, Universidade de Lisboa, 2018.

[3] "INESC-ID." https://www.inesc-id.pt/ (accessed Dec. 27, 2020).

[4] "APA SNIRH :: Sistema Nacional de Informação de Recursos Hídricos." https://snirh.apambiente.pt/index.php?idMain=5&idItem=5 (accessed Dec. 07, 2020).

[5] R. Gomes, M. Saramago, and R. Rodrigues, "Svarh – Sistema de Vigilância e Alerta de Recursos Hídricos," Technical Report, Instituto da Água, Lisboa, 2003.

[6] D. Conde, M. Telhado, M. Viana Baptista, and R. Ferreira, "Severity and exposure associated with tsunami actions in urban waterfronts: the case of Lisbon, Portugal," *Nat. Hazards*, vol. 79, no. 3, pp. 2125–2144, 2015, doi: 10.1007/s11069-015-1951-z.

[7] P. H. Carstensen and L. Vogelsang, "Design of Web-Based Information Systems - New Challenges for Systems Development?," in *Proceedings of Ecis*, 2001, pp. 536–547.

[8] A. R. da Silva, "ITLingo Research Initiative," Technical Report, Instituto Superior Técnico, Universidade de Lisboa, 2017.

[9] I. Gamito and A. Rodrigues da Silva, "From rigorous requirements and user interfaces specifications into software business applications," *Commun. Comput. Inf. Sci.*, vol. 1266 CCIS, pp. 459–473, 2020, doi: 10.1007/978-3-030-58793-2_37.

[10] D. Ince, *Object oriented design with applications*, 2nd ed., vol. 34, no. 9. Addison Wesley Longman, Inc., 1992.

[11] A. R. da Silva, "Model-driven engineering: A survey supported by the unified conceptual model," *Comput. Lang. Syst. Struct.*, vol. 43, pp. 139–155, 2015, doi: 10.1016/j.cl.2015.06.001.

[12] OGC, *OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture*. 2011, p. 93.

[13] D. Rubio, *Beginning Django*. Apress, 2017.

[14] G. Early, "Celebrating knowledge," *Proc. Am. Philos. Soc.*, vol. 151, p. 74, 2007.

[15] R. France and B. Rumpe, "Model-driven development of complex software: A research roadmap," *FoSE 2007 Futur. Softw. Eng.*, pp. 37–54, 2007, doi: 10.1109/FOSE.2007.14.

[16] A. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture : Practice and Promise*. 2003.

[17] S. P. Christodoulou and T. S. Papatheodorou, "Web Engineering Resources Portal (WEP): A Reference Model and Guide," *Web Eng. Princ. Tech.*, pp. 31–74, 2005.

[18] "IFML: The Interaction Flow Modeling Language | The OMG standard for front-end design." https://www.ifml.org/ (accessed Dec. 12, 2019).

[19] M. Brambilla and P. Fraternali, "Implementation of applications specified with IFML," *Interact. Flow Model. Lang.*, pp. 279–334, 2015, doi: 10.1016/b978-0-12-800108-0.00010-2.

[20] R. Pereira, "Técnicas Avançadas de Modelação e Produção Semi-Automática de Aplicações Web Responsivas," Instituto Superior Tecnico, Universidade de Lisboa, 2019.

[21] I. Gamito, "Automatic Production of Software Business Applications from Rigorous Requirements Specifications," Technical Report, Instituto Superior Técnico, Universidade de Lisboa, 2019.

[22] A. R. da Silva, A. C. R. Paiva, and V. E. R. da Silva, "Towards a test specification language for information systems: Focus on data entity and state machine tests," *Model. 2018 - Proc. 6th Int. Conf. Model. Eng. Softw. Dev.*, pp. 213–224, 2018, doi: 10.5220/0006608002130224.

[23] A. R. Silva, "Rigorous Requirements Specification with the RSL Language: Focus on Uses Cases," in *ISD'2019, AIS*, 2018, pp. 1–32.

[24] "Xtext - Language Engineering Made Easy!" https://www.eclipse.org/Xtext/ (accessed Dec. 10, 2020).

[25] "OGC." https://www.ogc.org/ (accessed Dec. 10, 2020).

[26] "OGC Standards." https://www.ogc.org/standards (accessed Dec. 10, 2020).

[27] "OGC® WaterML | OGC." https://www.ogc.org/standards/waterml#overview (accessed Oct. 27, 2020).

[28] "Coordinate Reference System and Spatial Projection | Earth Data Science - Earth Lab." https://www.earthdatascience.org/courses/earth-analytics/spatial-data-r/intro-to-coordinate-reference-systems/ (accessed Oct. 28, 2020).

[29] "Vector Data." https://docs.qgis.org/2.8/en/docs/gentle_gis_introduction/vector_data.html (accessed Dec. 10, 2020).

[30] "What is raster data?—Help | ArcGIS for Desktop." https://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/what-is-raster-data.htm (accessed Oct. 28, 2020).

[31] "Working with Mesh Data — QGIS Documentation

documentation." https://docs.qgis.org/3.10/en/docs/user_manual/working_with_mesh/mesh_properties.html#what-s-a-mesh (accessed Oct. 28, 2020).

[32] "The Web framework for perfectionists with deadlines | Django." https://www.djangoproject.com/ (accessed Dec. 10, 2020).

[33] "MVC Framework - Introduction - Tutorialspoint." https://www.tutorialspoint.com/mvc_framework/mvc_framework_introduction.htm (accessed Dec. 10, 2020).

[34] "GeoDjango | Django documentation | Django." https://docs.djangoproject.com/en/3.0/ref/contrib/gis/ (accessed Dec. 04, 2020).

[35] "GeoDjango Tutorial | Django documentation | Django." https://docs.djangoproject.com/en/3.0/ref/contrib/gis/tutorial/#introduction (accessed Dec. 10, 2020).

[36] "PostgreSQL: The world's most advanced open source database." https://www.postgresql.org/ (accessed Dec. 04, 2020).

[37] "PostGIS — Spatial and Geographic Objects for PostgreSQL." https://postgis.net/ (accessed Oct. 19, 2020).

[38] "GeoDjango Database API | Django documentation | Django." https://docs.djangoproject.com/en/3.0/ref/contrib/gis/db-api/ (accessed Dec. 04, 2020).

[39] "MIKE 2020." https://www.mikepoweredbydhi.com/mike-2020 (accessed Nov. 01, 2020).

[40] "MIKE 21." https://www.mikepoweredbydhi.com/products/mike-21 (accessed Nov. 01, 2020).

[41] M. Saramago, "Redes de Monitorização Hidrometeorológicas," *Rev. Recur. Hídricos, APRH*, vol. 38, no. 1, pp. 33–39, 2017.

[42] M. Saramago, "Hydrologic Surveillance System Using Wireless Technologies," 2006, [Online]. Available: http://projects.knmi.nl/geoss/ICEAWS/ICEAWS-4/CD/docs/ORAL/17_oral.pdf.

[43] "Mike 21 Flow Model & Mike 21 Flood Screening Tool." DHI, Horsholm, 2017, Accessed: Dec. 08, 2020. [Online]. Available: www.mikepoweredbydhi.com.

[44] DHI, "MIKE 21 Flow Model User Manual." DHI, p. 120, 2017.

[45] DHI, "Mike 21 Hd." DHI, 2011.

[46] M. Gonzalez, "RiverCure Portal : Collaborative GeoPortal for Curatorship of Digital Resources in the Water Management Domain," Instituto Superior Tecnico, Universidade de Lisboa, 2020.

[47] "GeoJSON." https://geojson.org/ (accessed Oct. 16, 2020).

[48] "Documentation | Visualizer." https://kitware.github.io/visualizer/docs/ (accessed Oct. 12, 2020).

[49] "Welcome to Flask — Flask Documentation (1.1.x)." https://flask.palletsprojects.com/en/1.1.x/foreword/ (accessed Dec. 10, 2020).

[50] "curl - Documentation Overview." https://curl.haxx.se/ (accessed Dec. 10, 2020).

[51] "ArcGIS Desktop | Documentation." https://desktop.arcgis.com/en/ (accessed Dec. 27, 2020).

[52] "Discover QGIS." https://www.qgis.org/en/site/about/index.html (accessed Dec. 10, 2020).